

Content Storage Format



- [Our Goals & Ideals](#)
 - [Storage Formats](#)
 - [Page Content](#)
 - [Images & Drawings](#)
 - [Attachments](#)
 - [Egress Options](#)
-

When building up a documentation store, or assessing platform options, it can be useful to understand how data is stored so you know how portable & accessible your content is. This page aims to clearly lay-out how content is stored within BookStack and what our general project aims are when it comes to data storage, content formats, and how these may lead design & development decisions.

Our Goals & Ideals

When it comes to core content provided into BookStack we want to avoid "locking" users into our platform. We believe data should be portable and to common standard formats, so that's what we aim for. For core page content we aim to stick to relatively simple HTML, with an aim to Commonmark for markdown where used (see ["page content" section](#) below for more details on this).

While we don't officially support export to (or import from) other specific platforms as part of the core project, we aim to provide options & guidance so this can be achieved where desired.

When assessing features & changes to BookStack, keeping content to simple standards in the interest of portability takes a very high level of priority in decisions. We have pushed back against many editor & content-format feature requests, in the interest of not complicating the content structure.

Storage Formats

BookStack is primarily a database-driven system. The vast majority of data and metadata can be found in tables of the attached database. For anything not specifically mentioned below, it's likely in the database. If your aim is BookStack instance backup/restore of any kind, then you should focus on performing a database dump (in addition to backing up files) as per our [backup/restore guidance here](#).

Page Content

Page content is primarily stored as HTML, with the `html` column of the `pages` database table. We aim (as per our goals) to keep the range of HTML formats limited to common basic HTML with little depth/structure complexity where possible. There's a few custom classes used (for alignment & callout blocks) but we now try to avoid the addition & use of new custom classes. Some formatting options may use inline HTML styles (Text color for example).

For pages written in Markdown, the original input markdown will be stored within the `markdown` column of the `pages` table. Within BookStack's official functions, we generally focus on standardising markdown support to [commonmark](#), with the extra additions of markdown tables, task-lists, and HTML.

Page content may reference other items within BookStack, and other external resources. In official functions, we aim to standardise on always using full absolute URLs, rather than any relative or custom dynamic references. This ensures such links/references work regardless of usage context, and having a full absolute URL provides a base URL/host that can be easily searched upon & detected where required.

There is one custom dynamic feature when it comes to page content, that being is our [dynamic include tag](#) system, which provides on-load inclusion of other content onto a page. Includes are not stored ready-parsed, they are handled at page display time since permissions can affect the result. Other than this, we avoid adding extra dynamic/"smart"/"magic" features to core page content.

Images & Drawings

Images are stored as standard image files, typically on the local filesystem but that can depend on [configured storage method](#). When uploading, image file names may be altered/generated by BookStack. Upon upload BookStack will store the originally provided image file data but also create & store resized images for convenience, mainly for reduced file size for more efficient loading & display. These system resized images are stored within directories that have names starting with `scaled-` or `thumbs-`, with these directories being in the same location as the original image files.

Drawings are treated much the same as images. When a drawing is saved in the integrated [diagrams.net](#) (Previously draw.io) editor, they're exported and saved within BookStack as standard PNG images. These PNG image files are embedded with the original drawing data, so they can be reloaded back into the diagrams.net to be fully editable again. You can drag/import these PNG files into any diagrams.net/draw.io instance for full re-editing capabilities.

Attachments

Attachments are stored as files, typically on the local filesystem but that can depend on [configured storage method](#). Filenames, including the file extension, will be altered so it may be hard to identify attachments on the filesystem by name. If you need to do this, you can use the `attachments` table of the database as a reference. The `path` column represents attachment file locations, relative to top-level storage location.

Egress Options

Note: This is not relevant for BookStack-only backup/restore operations, see [our guidance here](#) for that.

To get data out of BookStack (in bulk) there's two main ways:

- [The BookStack REST API](#)
- Fetch/export directly from the database.

The REST API presents a nice scriptable, primarily JSON-based, interface. Various example scripts can be found in [our api-scripts repo](#). The API covers all core content types, including their RAW underlying data. The API does provide access to export formats, but most of these may perform some transformation or be lossy in operation. The one exception may be the (contained) HTML export option since this will attempt to embed image content into the page which could make the content more portable, and potentially help avoid having to manage images separately.

Otherwise, directly interfacing with, or exporting from, the database is always an option. We attempt to have sensible table and column naming, with a simple overall database structure, so navigating around to extract/format data as required shouldn't be too much trouble if confident with database systems.

For either of these, some potential pain points for egress (migration away) from BookStack could be:

- Handling image/media/attachment content, and mapping/linking their use from content.
- Supporting any additional metadata you want to migrate.
- Handling any complex/custom elements of the content format.

The specifics & overall complexity can ultimately depend on what you need to migrate/transfer to, in addition to the skills & tools you have available.
